# Software Testing and Validation – 2017/18
## Instituto Superior Técnico

Prof. João Dias Pereira

# *Vos*
# Project Report

**Group 01 – Alameda**

**Francisca Cambra**
ist181057

**Rui Ventura**
ist181045

April 12, 2018

# Contents

# 1 Method-Scope Tests

## 1.1 `assignPhoneNumber`

Assigns a free phone number to a client of *Vos* if all conditions are met. If at least one does not hold, then this method does not change anything, throwing an `InvalidOperationException`.

### 1.1.1 Test Pattern – Category-Partition Test

### 1.1.2 Functions

- Primary function

    – Assign free phone number to a client without a number

- Secondary functions

    – Throw `InvalidOperationException` if conditions aren't met
        * Invalid nif (nif $\notin [10^8, 10^9[$)
        * A *Vos* client with the given nif doesn't exist
        * Invalid phone number (number $\notin [10^8, 10^9[$)
        * It isn't a *Vos* number
        * Phone number already assigned
        * Client can't be assigned any more numbers

### 1.1.3 Input/Output Parameters

- Input

    – `clientNif` – The nif of the client to assign a number to
    – `phoneNumber` – The phone number to be assigned
    – `clients` – The set of *Vos* clients managed by `ClientManager`

- Output

    – `client` – The updated client, if a number was assigned successfully

### 1.1.4 Categories & Choices

| Parameter | Category | Choices |
|---|---|---|
| `clientNif` | *Vos* client (w/ *#numbers* phone numbers) | $\#numbers \in [1, 5[$ |
| | | $\#numbers = 5$ (MAX) |
| | Not a *Vos* client | $\texttt{clientNif} \in [10^8, 10^9[$ |
| | Invalid nif | $\texttt{clientNif} \notin [10^8, 10^9[$ |
| `phoneNumber` | *Vos* phone number | Free (Unassigned) |
| | | Not free (Assigned) |
| | Not a *Vos* number | $\texttt{phoneNumber} \in [10^8, 10^9[$ |
| | Invalid number | $\texttt{phoneNumber} \notin [10^8, 10^9[$ |
| `clients` | *n*-elements | $n = 0$ (Empty) |
| | | $n \in [1, \text{MAX}]$ (Not empty) |

Table 1: Set of `assignPhoneNumber`'s input parameters broken into categories, accompanied by test case choices

### 1.1.5 Constraints

- Empty `clients` list precludes the assignment of a `phoneNumber` to a client (which, since the list is empty, mustn't exist)

- Assigning an invalid `phoneNumber`, one that doesn't belong to *Vos* or one that is already assigned is the same for any kind of client

### 1.1.6 Test Cases

| | Choices | | | Expected Result | |
|---|---|---|---|---|---|
| **TC** | clientNif | phoneNumber | clients | **Exception** | client |
| 1 | $\#numbers \in [1,5[$ | Free | $n \in [1, \text{MAX}]$ | NO | $\#numbers \in ]1,5]$ |
| 2 | $\#numbers \in [1,5[$ | Not free | $n \in [1, \text{MAX}]$ | YES | — |
| 3 | $\#numbers \in [1,5[$ | $\in [10^8, 10^9[$ | $n \in [1, \text{MAX}]$ | YES | — |
| 4 | $\#numbers \in [1,5[$ | $\notin [10^8, 10^9[$ | $n \in [1, \text{MAX}]$ | YES | — |
| 5 | $\#numbers = 5$ | Free | $n \in [1, \text{MAX}]$ | YES | — |
| 6 | $\in [10^8, 10^9[$ | Free | $n \in [1, \text{MAX}]$ | YES | — |
| 7 | $\notin [10^8, 10^9[$ | Free | $n \in [1, \text{MAX}]$ | YES | — |

Table 2: Set of reduced test cases for the `assignPhoneNumber` method after constraints are applied

## 1.2 `computeBill` method

The responsibility of `computeBill` method is to determine the value to pay for a client taking into account all communications made by the client through all of his registered mobile phones

### 1.2.1 Test Pattern – Combinational Function Test

### 1.2.2 Decision Tree
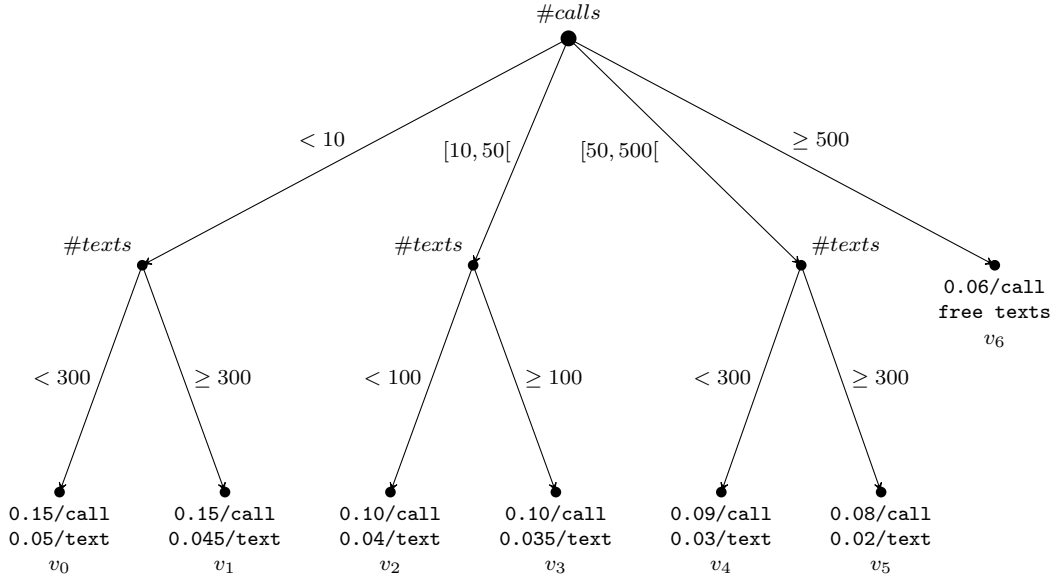


Figure 1: Decision tree describing the ouput given by `computeBill` based on the number of texts sent and calls made by the client

### 1.2.3 Domain Matrices

| $v_0$ | | | Test Cases | | | |
|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | — | 1 | — | 2 |
| $\#calls$ | $< 10$ | ON | 10 | | | |
| | | OFF | | 9 | | |
| | Typical | IN | | | 8 | 7 |
| $\#texts$ | $< 300$ | ON | | | 300 | |
| | | OFF | | | | 299 |
| | Typical | IN | 147 | 204 | | |
| **Expected Result** | | | $v_3$ | 11.55 | $v_1$ | 16.00 |

Table 3: $v_0$ domain matrix

| $v_1$ | | | Test Cases | | | |
|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | — | 3 | 4 | — |
| *#calls* | $< 10$ | ON | 10 | | | |
| | | OFF | | 9 | | |
| | Typical | IN | | | 6 | 5 |
| *#texts* | $\geq 300$ | ON | | | 300 | |
| | | OFF | | | | 299 |
| | Typical | IN | 320 | 400 | | |
| **Expected Result** | | | $v_3$ | 19.35 | 14.40 | $v_0$ |

Table 4: $v_1$ domain matrix

| $v_2$ | | | Test Cases | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | 5 | — | — | 6 | — | 7 |
| *#calls* | $\geq 10$ | ON | 10 | | | | | |
| | | OFF | | 9 | | | | |
| | $< 50$ | ON | | | 50 | | | |
| | | OFF | | | | 49 | | |
| | Typical | IN | | | | | 22 | 35 |
| *#texts* | $< 100$ | ON | | | | | 100 | |
| | | OFF | | | | | | 99 |
| | Typical | IN | 48 | 20 | 33 | 15 | | |
| **Expected Result** | | | 2.92 | $v_0$ | $v_4$ | 5.50 | $v_3$ | 7.46 |

Table 5: $v_2$ domain matrix

| $v_3$ | | | Test Cases | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | 8 | — | — | 9 | 10 | — |
| *#calls* | $\geq 10$ | ON | 10 | | | | | |
| | | OFF | | 9 | | | | |
| | $< 50$ | ON | | | 50 | | | |
| | | OFF | | | | 49 | | |
| | Typical | IN | | | | | 12 | 44 |
| *#texts* | $\geq 100$ | ON | | | | | 100 | |
| | | OFF | | | | | | 99 |
| | Typical | IN | 148 | 220 | 333 | 414 | | |
| **Expected Result** | | | 6.18 | $v_0$ | $v_5$ | 15.49 | 4.70 | $v_2$ |

Table 6: $v_3$ domain matrix

| $v_4$ | | | Test Cases | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | 11 | — | — | 12 | — | 13 |
| *#calls* | $\geq 50$ | ON | 50 | | | | | |
| | | OFF | | 49 | | | | |
| | $< 500$ | ON | | | 500 | | | |
| | | OFF | | | | 499 | | |
| | Typical | IN | | | | | 142 | 51 |
| *#texts* | $< 300$ | ON | | | | | 300 | |
| | | OFF | | | | | | 299 |
| | Typical | IN | 240 | 189 | 98 | 10 | | |
| **Expected Result** | | | 11.70 | $v_3$ | $v_6$ | 45.21 | $v_5$ | 13.56 |

Table 7: $v_4$ domain matrix

| $v_5$ | | | Test Cases | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | 14 | — | — | 15 | 16 | — |
| *#calls* | $\geq 50$ | ON | 50 | | | | | |
| | | OFF | | 49 | | | | |
| | $< 500$ | ON | | | 500 | | | |
| | | OFF | | | | 499 | | |
| | Typical | IN | | | | | 200 | 60 |
| *#texts* | $\geq 300$ | ON | | | | | 300 | |
| | | OFF | | | | | | 299 |
| | Typical | IN | 314 | 500 | 616 | 404 | | |
| **Expected Result** | | | 10.28 | $v_3$ | $v_6$ | 48.00 | 22.00 | $v_4$ |

Table 8: $v_5$ domain matrix

| $v_6$ | | | Test Cases | |
|---|---|---|---|---|
| **Variable** | **Condition** | **Type** | 17 | — |
| *#calls* | $\geq 500$ | ON | 500 | |
| | | OFF | | 499 |
| **Expected Result** | | | 30.00 | $v_4/v_5$ |

Table 9: $v_6$ domain matrix

## 2  Class-Scope Tests

### 2.1  `Client` class

Each client of *Vos* has a name (with a minimal length of 5) and by its social security number (designated as nif). This number is a unique identifier in *Vos*. A client can have several phone numbers managed by *Vos* (between 1 and 5). Each client can associate a mobile phone to each of his assigned phone numbers.

Each client can register in the system a given amount of phone number of *friends*. The maximum number of phone number a client can register is equal to three times the number of phone numbers plus five.

#### 2.1.1  Test Pattern – Non-Modal Class Test

#### 2.1.2  Class Invariant

| Client variables | |
|---|---|
| **Variable** | **Type** |
| name | String |
| nif | int |
| numbers | List<Integer> |
| friends | List<Integer> |

Table 10: `Client` class' variables and their respective types

Domain restrictions

- `name.length()` $\geq 5$

- `nif` $\in [10^8, 10^9[$

- `numbers.size()` $\in [1, 5]$

- `friends.size()` $\leq 3 \times$ `numbers.size()` $+ 5$

The logical conjunction of all of these restrictions makes up the Class Invariant

#### 2.1.3  On and Off points

| Boundary | ON | OFF |
|---|---|---|
| `name.length()` $\geq 5$ | 5 | 4 |
| `nif` $\geq 10^8$ | $10^8$ | $10^8 - 1$ |
| `nif` $< 10^9$ | $10^9$ | $10^9 - 1$ |
| `numbers.size()` $\geq 1$ | 1 | 0 |
| `numbers.size()` $< 5$ | 5 | 4 |
| `friends.size()` $\leq 3n^1 + 5$ | $3n + 5$ | $3n + 6$ |

Table 11: On and Off points for the `Client` class' invariant boundaries

---

[1] `numbers.size()`

## 2.1.4  Domain Matrix

| Variable | Condition | Type | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Boundary** | | | | | | | | | **Test Cases** | | | | | |
| name.length() | $\geq 5$ | ON | 5 | | | | | | | | | | | |
| | | OFF | | 4 | | | | | | | | | | |
| | Typical | IN | | | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| nif | $\geq 10^8$ | ON | | | $10^8$ | | | | | | | | | |
| | | OFF | | | | $10^8 - 1$ | | | | | | | | |
| | $< 10^9$ | ON | | | | | $10^9$ | | | | | | | |
| | | OFF | | | | | | $10^9 - 1$ | | | | | | |
| | Typical | IN | $10^8 + 1$ | $10^8 + 2$ | | | | | $10^8 + 3$ | $10^8 + 4$ | $10^8 + 5$ | $10^8 + 6$ | $10^8 + 7$ | $10^8 + 8$ |
| numbers.size() | Typical | IN | 2 | 3 | 4 | 3 | 2 | 3 | | | | | 4 | 3 |
| | $\geq 1$ | ON | | | | | | | 1 | | | | | |
| | | OFF | | | | | | | | 0 | | | | |
| | $< 5$ | ON | | | | | | | | | 5 | | | |
| | | OFF | | | | | | | | | | 4 | | |
| friends.size() | Typical | IN | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
| | $\leq 3n + 5$ | ON | | | | | | | | | | | 17 | |
| | | OFF | | | | | | | | | | | | 15 |
| **Expected Result** | Typical | IN | Y | N | Y | N | N | Y | Y | N | N | Y | Y | N |

Table 12: Client class test cases

## 2.2 `Mobile` class

A mobile phone can make and receive calls and send and receive texts. A mobile phone can be turned on or off (and in this case it cannot make calls, send texts and receive calls nor texts). It has two modes (*friend* and *silent*) that can be enabled or disabled.

### 2.2.1 Test Pattern – Modal Class Test
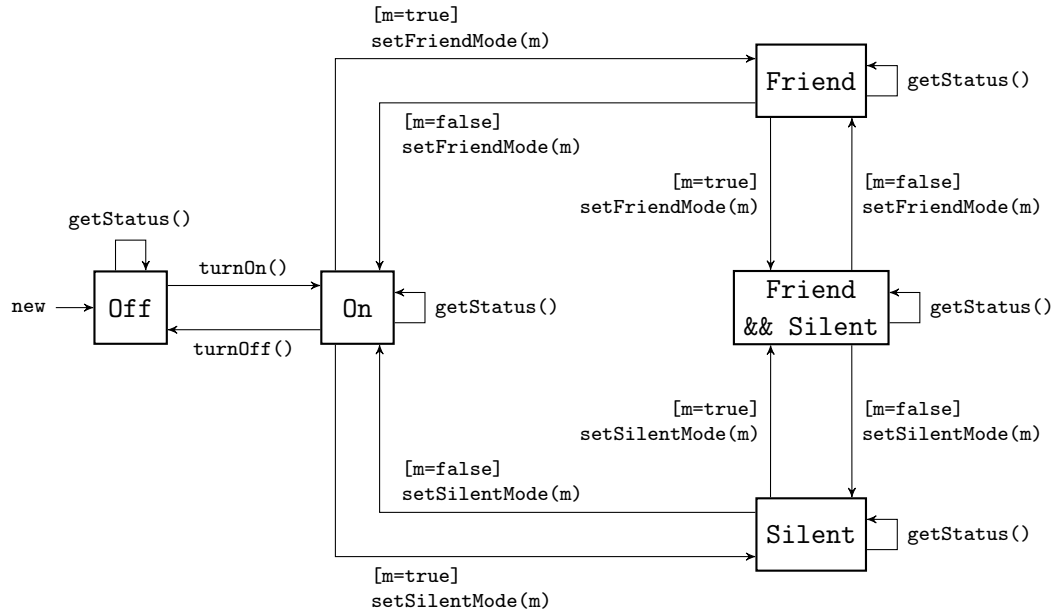
### 2.2.2 Finite State Machine



Figure 2: `Mobile` class state machine, representing the class' states and transitions between them